REMARKS

Claims 1-4 and 7-27 are pending, of which claim 1 is an independent system claim, claim 4 is an independent method claim, with a corresponding independent computer program product claim 15, and claim 24 is an independent method claim. As shown above, independent claims 1, 4, 15, and 24 have been amended by this paper, and dependent claims 7, 18-21, and 27 have been amended to use language that is consistent with the amendments made to the corresponding independent claim.[1]

The Office Action rejected each of the pending independent claims (1, 4, 15, and 24) and all dependent claims, except claim 3, under 35 U.S.C. § 102(b) as being anticipated by TETware release 3.3 ("TETware") as described in the TETware User Guide, Revision 1.2 ("TETware UG") and the TETware Programmers Guide, Revision 1.2 ("TETware PG"). The Office Action rejected claim 3 under 35 U.S.C. § 103(a) as being unpatentable over TETware in view of U.S. Patent No. 6,505,342 to Hartmann et al. ("*Hartmann*").[2]

Applicants' invention, as recited for example in independent system claim 1, relates to a computer system for selecting and organizing individual test cases from a binary program module of test cases for use in testing a computer program to ensure that the program processes as intended. The system includes a binary program module storing a plurality of individually accessible test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface; a harness comprising a set of instructions that executes one or more selected test cases within a test case hierarchy on the computer program using the corresponding language and format independent interface of the one or more selected test cases; a connector comprising a set of instructions that scans the program module and extracts the one or more available test cases for use in testing the computer program to ensure that it processes as intended, the connector creating a hierarchy of the one or more available test cases, and selectively integrating an interface between the test case hierarchy and the harness regardless of the language or format in which the one or more available test cases were written; a harness client comprising a set of instructions that (i) receives user input for

[1]Support for the Amendments can be found throughout the Specification, and particularly beginning line 22 of page 12.

[2]Although the prior art status of the cited art is not being challenged at this time, Applicants reserve the right to do so in the future. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status or asserted teachings of the cited art.

specifying a search property for identifying the one or more available test cases to be the executed on the computer program, (ii) employs the connector to create the test case hierarchy of the one or more available test cases, (iii) receives user input for selecting the one or more selected test cases from the one or more available test cases to execute on the computer program, and (iv) receives user input for specifying how the one or more selected test cases are to be executed on the computer program; and a processor for executing the one or more selected test cases, the harness and the connector.

Applicants' invention, as recited for example in independent method claim 4, relates to selecting and extracting test cases from a binary program module for use in testing a computer program to determine whether the computer program processes as intended. The method includes a harness client receiving user input that (i) specifies a search property to identify one or more test cases of interest, (ii) selects one or more test cases from the one or more test cases of interest to execute on the computer program, and (iii) specifies how the one or more selected test cases are to be executed on the computer program; a connector scanning the binary program module storing the plurality of individually accessible test cases for one or more test cases of interest, each test case having a language and format independent interface for executing the test case on the computer program regardless of the language or format used to develop the test case; the connector extracting the one or more test cases of interest from the binary program module; the connector organizing one or more test cases into a test case hierarchy; the connector interfacing a harness with the one or more test cases of interest, wherein the interfacing allows the harness to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed; and the harness traversing the test case hierarchy and executing each of the one or more selected test cases to test the computer program. Independent claim 15 recites similar limitations from the perspective of a computer program product.

Applicants' invention, as recited for example in independent method claim 24, similarly relates to testing a computer program to determine whether the computer program processes as intended. The method includes receiving user input that (i) specifies a search property for identifying one or more test cases, (ii) selects one or more test cases from the one or more test cases of interest to execute on the computer program, and (iii) specifies how the one or more selected test cases are to be executed on the computer program; identifying one or more test

cases from a binary program module that stores a plurality of individually accessible test cases based on the search property each test case implementing a language and format independent interface for executing the test case on a computer program regardless of the language or format used to develop the test case; translating the identified one or more test cases into a test case hierarchy; interfacing the test case hierarchy in order to recognize and execute the one or more test cases regardless of the language or format in which the one or more test cases were written; and executing each of the one or more selected test cases in the test case hierarchy to test the computer program regardless of the language or format in which the one or more test cases were written.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." MPEP § 2131. That is, "for anticipation under 35 U.S.C. 102, the reference must teach every aspect of the claimed invention either explicitly or impliedly." MPEP § 706.02. Applicants also note that "[i]n determining that quantum of prior art disclosure which is necessary to declare an applicant's invention 'not novel' or 'anticipated' within section 102, the stated test is whether a reference contains an 'enabling disclosure.'" MPEP § 2121.01. In other words, a cited reference must be enabled with respect to each claim limitation. During examination, the pending claims are given their broadest reasonable interpretation, *i.e.*, they are interpreted as broadly as their terms reasonably allow, consistent with the specification. MPEP §§ 2111 & 2111.01.

As noted in Applicants' prior response, TETware discloses grouping test cases within a test suite. TETware PG, section 2.2. Test suites are organized as directory hierarchies—the top of each test suite directory is known as the test suite root directory. TETware UG, section 5.2.6. All files in a test suite reside below the test suite directory (or in a specified alternate execution directory). TETware PG, section 2.3; TETware UG, section 5.2.6. Test suites are required to include certain files and utilities, such as a build tool (e.g., make), a clean tool (e.g., rm), at least one test scenario file, etc. TETware PG, section 2.5.

A test scenario is a list of invocable components from a test suite that are processed during a particular TETware invocation. TETware UG, section 2.2. Within a scenario file, a test case name may appear by itself or be attached to a directive that describes how the test case should be executed (sequentially, in parallel with other test cases, repetitively, remotely, etc.). TETware PG, section 4.2.4.3; TETware UG, section 5.3.2.2. Test case names are interpreted

relative to the test suite root directory or alternate execution directory, depending on the mode of operation. TETware UG, section 5.3.2.4. Section 5.3.2.5 of the TETware UG and section 4.4 of the TETware PG present some simple examples of test scenarios. Example test cases names listed in these scenarios follow the directory convention explained above (e.g. "/tset/tc1" and "/ts/tc1").

The test cases in a test suite are processed by a Test Case Controller (tcc), based on a chosen mode of operation (e.g., build, execute, clean up). TETware PG, section 2.5. In build mode, the tcc translates source test cases into executables, in execute mode the tcc loads and executes test cases, and in clean mode the tcc removes unwanted files. TETware PG, section 3.2. Each test case is an executable program. TETware PG, section 2.2. When a test case uses one of the TETware language specific APIs, its execution is supervised by a Test Case Manager (TCM). TETware PG, section 2.4.2. The TCM is not a separate program, but instead is linked with user-supplied test code and the API library to produce an executable test case. *Id.* There is a separate TCM module for each API that is supported by TETware. *Id.* For example, TETware includes a C TCM and a C++ TCM. TETware PG, section 2.4. Test cases are written to a specific language binding (C, C++, Shell, Korn Shell, etc.). TETware PG, sections 8, 9, 10, and 11.

Among other things, however, TeTWare fails to teach, suggest, or enable a harness client comprising a set of instructions that (i) receives user input specifying a search property for identifying the one or more available test cases to be the executed on the computer program, (ii) employs the connector to create the test case hierarchy of the one or more available test cases, (iii) receives user input for selecting the one or more selected test cases from the one or more available test cases to execute on the computer program, and (iv) receives user input for specifying how the one or more selected test cases are to be executed on the computer program, as recited in independent claim 1, and fails to teach, suggest, or enable receiving user input that (i) specifies a search property to identify one or more test cases of interest, (ii) selects one or more test cases from the one or more test cases of interest to execute on the computer program, and (iii) specifies how the one or more selected test cases are to be executed on the computer program, as recited in independent claims 4, 15, and 24.

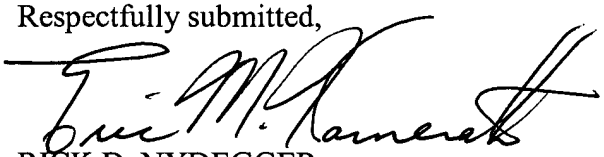
Based on at least the foregoing reasons, Applicants respectfully submit that the cited prior art fails to anticipate or make obvious Applicants invention, as claimed for example, in

independent claims 1, 4, 15, and 24. Applicants note for the record that the remarks above render the remaining rejections of record for the independent and dependent claims moot, and thus addressing individual rejections or assertion with respect to the teachings of the cited art is unnecessary at the present time, but may be undertaken in the future if necessary or desirable, and Applicants reserve the right to do so.

In the event that the Examiner finds any remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney.

Dated this 16th day of September, 2004.

Respectfully submitted,



RICK D. NYDEGGER
Registration No. 28,651
ERIC M. KAMERATH
Registration No. 46,081
Attorneys for Applicant
Customer No. 022913

EMK:kc
KC0000002704V001